BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Berikut merupakan penjelasan metodologi penelitian yang dilakukan untuk penelitian ini, beberapa metodologi akan dijelaskan lebih detail pada subbab-subbab berikutnya:

1. Pengumpulan Data *Game*

Data Game dikumpulkan melalui API berbasis HTTP yang mengembalikan data berbentuk JSON atau HTML untuk di-*crawl* dan *package* Steam untuk python yang disimpan ke dalam *database*.

2. Studi Literatur

Studi literatur dilakukan untuk memahami teori-teori yang akan digunakan selama perancangan dan pembangunan aplikasi. Teori-teori yang dimaksud antara lain: *Artificial Intelligence*, *Machine Learning*, *Supervised Learning/Unsupervised Learning*, *Decision Tree Learning*, *Bootstrap Aggregation*, dan *Random Forest*.

3. Proses Data *Game*

Data *game* yang sudah disimpan pada *database*, diproses untuk menentukan pendekatan apa saja yang dapat dilakukan, seperti rasio *dataset*, parameter yang digunakan untuk pelatihan, dan memilah data untuk dijadikan *dataset* yang akan digunakan untuk pelatihan dan pengujian.

4. Perancangan Aplikasi

Perancangan aplikasi meliputi *flowchart* aplikasi, struktur *database* data *game*, skema *database* data *game*, desain antarmuka aplikasi dan diagram infrastruktur aplikasi.

5. Pembangunan Aplikasi

Aplikasi dibangun dengan basis *web* yang menggunakan Bahasa pemrograman PHP dan *framework* PHP CodeIgniter. Data *game* disimpan pada *database* MySQL sebagai tempat penyimpanan lokal untuk melakukan pelatihan dan pengujian pembelajaran mesin. Untuk pelaksanaan pembelajaran mesin, aplikasi menggunakan bahasa pemrograman Python dan *package* scikit-learn dari https://scikit-learn.org/.

6. Pengujian Aplikasi

Aplikasi akan diuji *accuracy*, *precision*, *recall* dan *f1-score* untuk model klasifikasi dan akan diuji *mean absolute error* dan *root mean square error* untuk model regresi.

7. Penulisan Laporan

Penulisan laporan berisikan seluruh aktivitas penelitian, seluruh hasil penelitian, dan kesimpulan dari dilakukannya penelitian.

3.2 Pengumpulan Data Game

Data *game* dikumpulkan dengan melakukan *query* pada URL sudah disediakan oleh Steam dan Steam Charts. Untuk daftar semua *game* membutuhkan Steam Web API Key untuk dapat melakukan *query*, API Key bisa didapatkan melalui halaman https://steamcommunity.com/dev/apikey dan *login* menggunakan

akun Steam yang bisa didaftarkan secara gratis. Data yang didapat dari *query* berbentuk JavaScript Object Notation (JSON), JSON adalah sebuah format untuk pertukaran data yang ringan, berbasis teks, dan *language-independent syntax* (Ecma International, 2017).

Penjelasan URL, parameter yang digunakan, dan data yang didapat sebagai berikut:

1. Daftar semua *game*

   URL: https://api.steampowered.com/IStoreService/GetAppList/v1/

   Parameter:

   a. key=<key>, dimana <key> adalah Steam Web API Key.

   b. include_games=1

   c. include_dlc=0

   d. include_software=0

   e. include_videos=0

   f. include_hardware=0

   g. max_results=100000

   data yang didapat adalah JSON yang berisikan daftar appid dan nama *game*.

2. Data detail *game*

   URL: https://store.steampowered.com/api/appdetails/

   Parameter:

   a. appids=<appid>

   data yang didapat adalah JSON yang berisikan detail *game* tersebut yaitu: appid, batasan umur, gratis atau tidak, dukungan *controller*, harga dalam rupiah,

platform *game* dirilis, tanggal *game* dirilis, *developer* dan *publisher*, dan bahasa yang didukung.

3. Data ulasan *game*

   URL: https://store.steampowered.com/appreviews/<appid>

   Parameter:

   a. json=1

   b. filter=all

   c. purchase_type=all

   d. review_type=all

   e. language=all

   data yang didapat adalah JSON yang berisikan appid, jumlah ulasan positif, jumlah ulasan negatif, dan jumlah total ulasan.

4. Data jumlah pemain *game*

   Untuk jumlah pemain *game*, data diambil dari Steam Charts dengan melakukan *crawling* tabel halaman web.

   URL: https://steamcharts.com/app/<appid>

   data yang didapat adalah HTML halaman *game* tersebut yang berisikan *average player* dan *peak player* per bulannya sejak Juli 2012 atau *game* tersebut dirilis.

Untuk data Steam Tags *game*, didapatkan melalui *package* Steam untuk python oleh ValvePython dari https://github.com/ValvePython/steam dengan membuat *script* python yang melakukan interaksi Steam Client untuk mendapatkan Steam Tags dari *game* tersebut. Untuk daftar Steam Tags, dilakukan *query* SQL

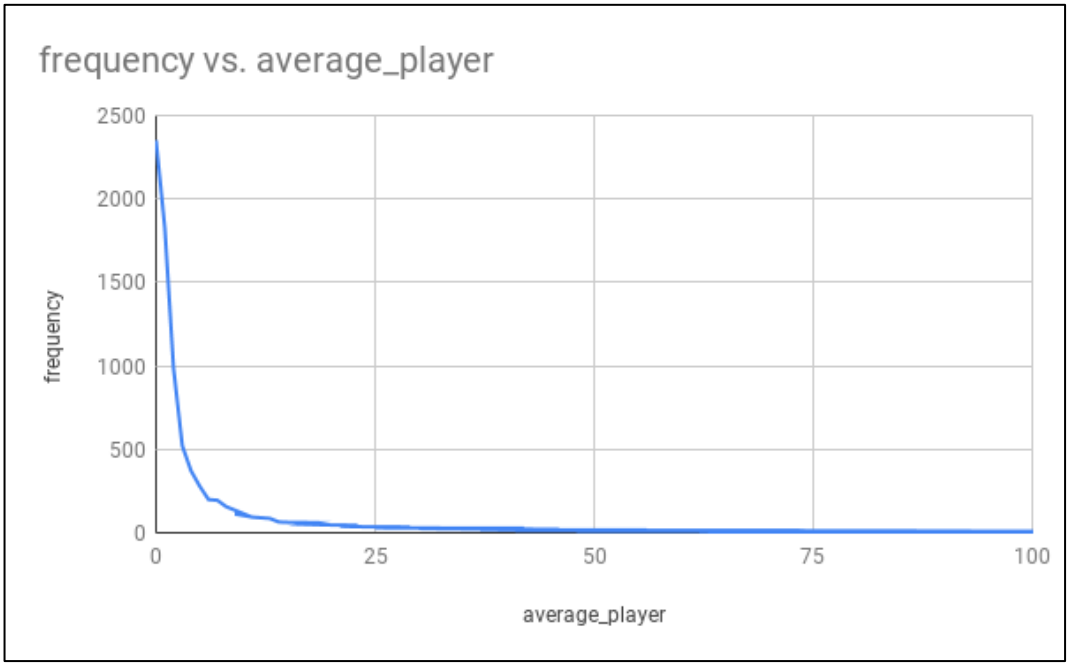
untuk Steam Tags *game* dengan DISTINCT untuk menampilkan tagid secara unik dan di-*cross-check* dengan daftar Steam Tags dari SteamDB.

3.3 Proses Data Game

Data *game* yang sudah tersimpan di *database* akan diproses sesuai dengan kebutuhan skenario-skenario percobaan implementasi. Berikut merupakan penjelasan pemrosesan data menjadi *dataset* untuk pelatihan dan pengujian.

3.3.1 Data Jumlah Pemain

Data jumlah pemain adalah data yang ingin diprediksi oleh aplikasi, namun data yang digunakan tidak seimbang/berat sebelah, maka dari itu digunakan juga alternatif lain selain jumlah pemain untuk data yang ingin diprediksi dan masih ada relevansi dengan jumlah pemain yaitu data jumlah ulasan. Gambar 3.1 memperlihatkan data yang tidak seimbang yang padahal mempunyai rentang data sampai jumlah *average player* 426,080.



Gambar 3.1 Frekuensi *average player* November 2019 (0 sampai 100)

3.3.2 Rasio Dataset

Rasio *dataset* dijadikan sebagai parameter penyetelan saat melakukan pelatihan untuk mencari skenario terbaik. Rasio *dataset* yang digunakan adalah sebagai berikut: 60% pelatihan 40% pengujian, 70% pelatihan 30% pengujian, dan 80% pelatihan 20% pengujian.

3.3.3 Fitur Dataset

Data yang digunakan untuk fitur *dataset* berdasarkan data yang sudah berbentuk *boolean* atau dapat diubah menjadi deretan *boolean*, data tersebut adalah gratis atau tidak, platform *game* dirilis, bulan *game* dirilis, bahasa yang didukung, dan Steam Tags *game*. Untuk bahasa yang didukung, akan menggunakan 30 bahasa teratas dan 10 bahasa teratas, dan untuk Steam Tags *game*, akan menggunakan 80% Steam Tags teratas dan 20 Steam Tags teratas.

3.3.4 Penyaringan Data

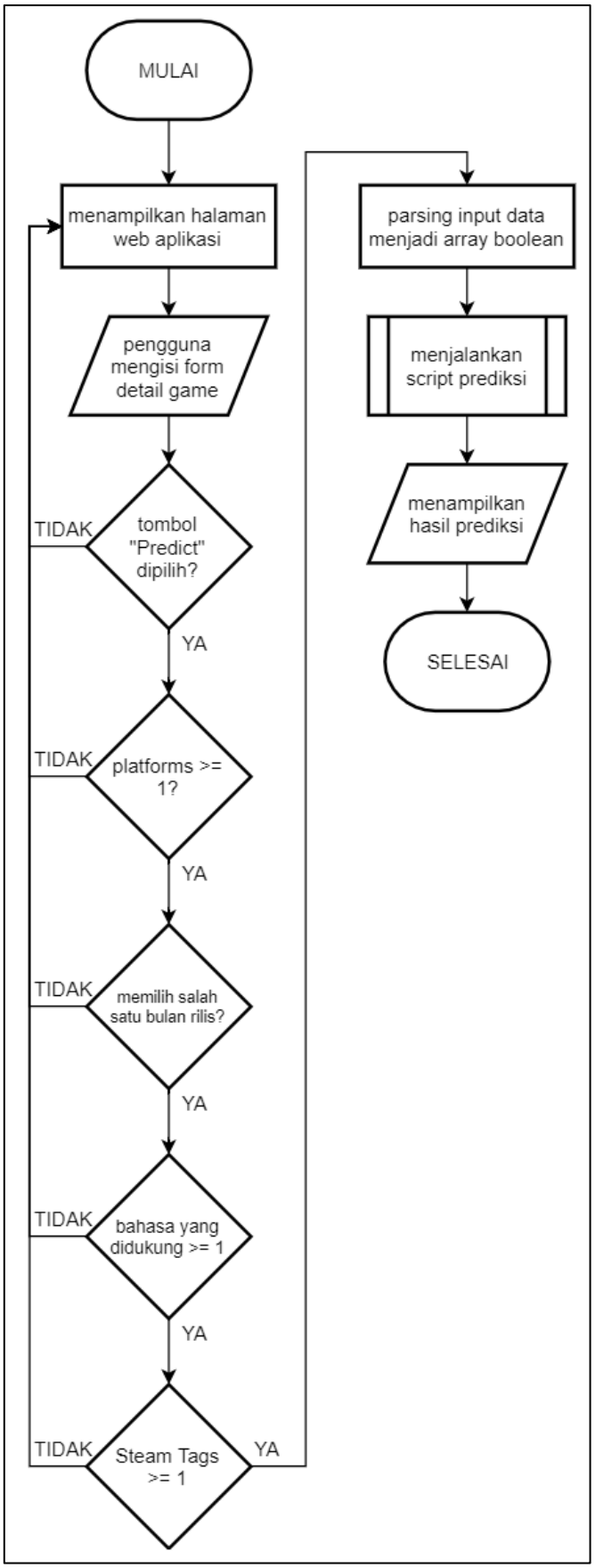
Menurut Aguinis dkk (Aguinis, et al., 2013), ada 20 *outlier handling techniques* yang bisa dilakukan untuk menangani *outlier* pada *dataset*, seperti *truncation*, *winsorization*, *transformation*, dan lain-lain. Salah satu *outlier handling techniques* yang digunakan adalah membuang data *outlier* dari *dataset*. Penyaringan data dilakukan juga untuk membuang data yang tidak memenuhi kriteria untuk dijadikan *dataset*, seperti tanggal rilis yang tidak lengkap. Penyaringan data membuang data dengan persentil <5% dan >95% yang bertujuan untuk membuang data *outlier* dari *dataset*.

3.4 Perancangan Aplikasi

Berikut merupakan penjabaran *flowchart* aplikasi, struktur *database* data *game*, dan desain antarmuka aplikasi.

3.4.1 Flowchart Aplikasi Web

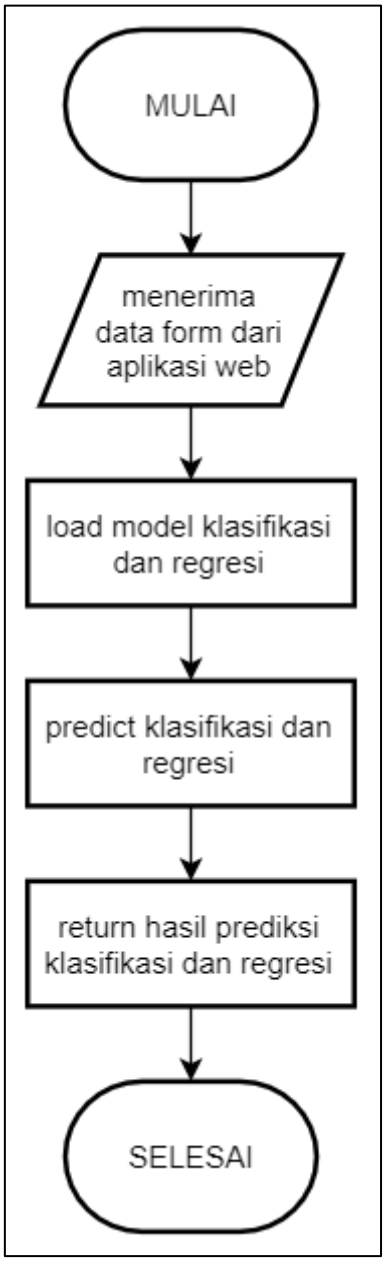
Gambar 3.2 merupakan *flowchart* aplikasi web. Aplikasi web menampilkan halaman web yang berisikan *form* yang akan diisi oleh user mengenai *game* yang ingin dibuat, lalu aplikasi web akan mengecek kriteria *form* apakah mencukupi untuk bisa menjalankan *script* prediksi, jika mencukupi maka *script* prediksi akan dijalankan untuk memprediksi jumlah pemain dan menampilkan hasil prediksi.



Gambar 3.2 Flowchart Aplikasi Web

3.4.2 Flowchart Script Prediksi

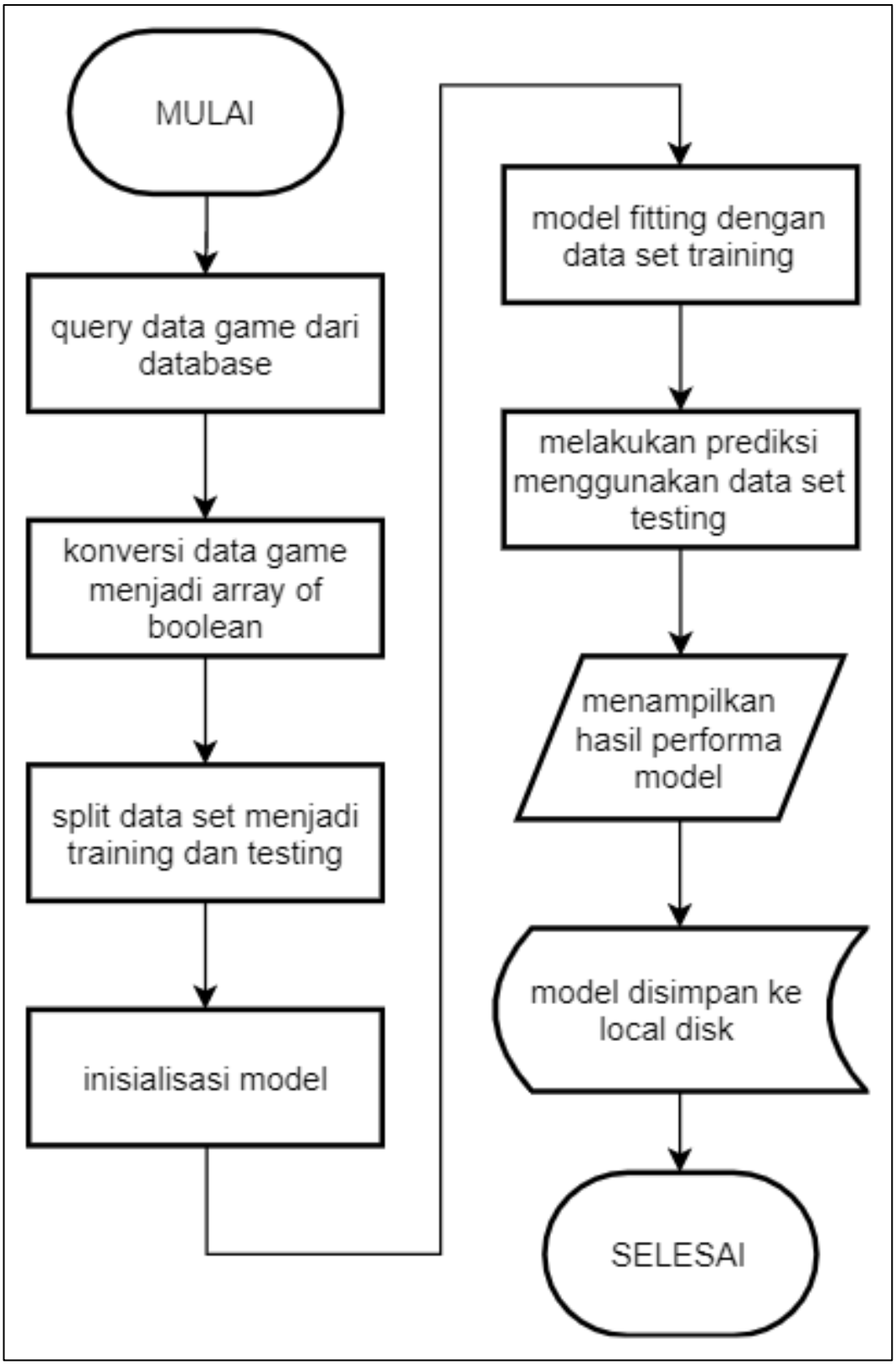
Gambar 3.3 merupakan *flowchart script* prediksi. *Script* menerima data dari *form* aplikasi web, me-*load* model klasifikasi dan regresi yang sudah tersimpan, melakukan prediksi dengan model yang sudah di-*load* dan data yang diberikan, dan mengembalikan hasil prediksi ke aplikasi web untuk ditampilkan.



Gambar 3.3 Flowchart Script Prediksi

3.4.3 Flowchart Script Training

Gambar 3.4 merupakan *flowchart script training*. *Script* akan melakukan *query* data *game* dari *database* sesusai dengan kriteria, mengonversi data *game* menjadi *dataset* berbentuk deretan *boolean*, membagi *dataset* menjadi *dataset* pelatihan dan *dataset* pengujian, model diinisialisasi sesuai dengan parameter-parameter yang ada, model melakukan *fitting* dengan *dataset* pelatihan, model yang sudah di-*fit* akan melakukan prediksi menggunakan *dataset* pengujian, lalu akan ditampilkan hasil performa model berdasarkan hasil prediksi dengan nilai sebenarnya, model akan disimpan kedalam *local disk*.



Gambar 3.4 Flowchart Script Training

3.4.4 Struktur Tabel game_details

Tabel 3.1 adalah struktur table game_details yang berfungsi untuk menyimpan data detail seluruh *game*.

Tabel 3.1 Struktur Tabel game_details

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Foreign Key* ke tabel steam_games.appid
type	tinytext	-	tipe objek pada katalog Steam
required_age	tinyint unsigned	3	minimal umur untuk memainkan *game*
is_free	binary	1	penanda *game* gratis atau tidak
controller_support	tinytext	-	informasi dukungan *controller*, antara lain ‘full’ dan ‘no or partial support’
price_idr	int unsigned	10	harga *game* dalam rupiah
platform_windows	binary	1	penanda *game* dirilis mendukung platform windows
platform_mac	binary	1	penanda *game* dirilis mendukung platform mac
platform_linux	binary	1	penanda *game* dirilis mendukung platform linux
coming_soon	binary	1	penanda *game* belum dirilis
steam_release_date	date *NULL*	-	tanggal *game* dirilis di Steam

3.4.5 Struktur Tabel game_developers

Tabel 3.2 adalah struktur table game_developers yang berfungsi untuk menyimpan data nama *developer game*.

Tabel 3.2 Struktur Tabel game_developers

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Foreign Key* ke tabel steam_games.appid
developers_order	tinyint unsigned	3	nomor urutan nama *developer*
developers_name	text *NULL*	-	nama *developer*

3.4.6 Struktur Tabel game_players

Tabel 3.3 adalah struktur tabel game_players yang berfungsi untuk menyimpan data jumlah pemain.

Tabel 3.3 Struktur Tabel game_players

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Foreign Key* ke tabel steam_games.appid
month_year	date	-	bulan dan tahun data jumlah pemain terhitung
average_player	int unsigned	10	jumlah pemain rata-rata pada bulan tersebut
peak_players	int unsigned	10	jumlah pemain tertinggi pada bulan tersebut

3.4.7 Struktur Tabel game_publishers

Tabel 3.4 adalah struktur tabel game_publishers yang berfungsi untuk menyimpan data nama *publisher game*.

Tabel 3.4 Struktur Tabel game_publishers

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Foreign Key* ke tabel steam_games.appid
publishers_order	tinyint unsigned	3	nomor urutan nama *publisher*
publishers_name	text *NULL*	-	nama *publisher*

3.4.8 Struktur Tabel game_reviews

Tabel 3.5 adalah struktur tabel game_reviews yang berfungsi untuk menyimpan data jumlah ulasan *game*.

Tabel 3.5 Struktur Tabel game_reviews

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Foreign Key* ke tabel steam_games.appid
total_positive	int unsigned	11	jumlah ulasan positif
total_negative	int unsigned	11	jumlah ulasan negatif
total_reviews	int unsigned	11	jumlah total ulasan

3.4.9 Struktur Tabel game_supported_languages

Tabel 3.6 adalah struktur tabel game_supported_languages yang berfungsi untuk menyimpan data bahasa yang didukung oleh *game*.

Tabel 3.6 Struktur Tabel game_supported_languages

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Foreign Key* ke tabel steam_games.appid
supported_language	tinytext	-	nama bahasa yang didukung

3.4.10 Struktur Tabel game_tags

Tabel 3.7 adalah struktur tabel game_tags yang berfungsi untuk menyimpan data Steam Tags *game*.

Tabel 3.7 Struktur Tabel game_tags

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Foreign Key* ke tabel steam_games.appid
tagid	int unsigned	10	*Foreign Key* ke tabel steam_tags.tagid

Tabel 3.7 Struktur Tabel game_tags (lanjutan)

Nama Kolom	Tipe	Panjang	Keterangan
tagid_order	int unsigned	10	nomor urutan Steam Tags

3.4.11 Struktur Tabel steam_games

Tabel 3.8 adalah struktur tabel steam_games yang berfungsi untuk menyimpan data daftar seluruh *game*.

Tabel 3.8 Struktur Tabel steam_games

Nama Kolom	Tipe	Panjang	Keterangan
appid	int unsigned	10	*Primary Key*
game_name	text	-	nama *game*

3.4.12 Struktur Tabel steam_tags

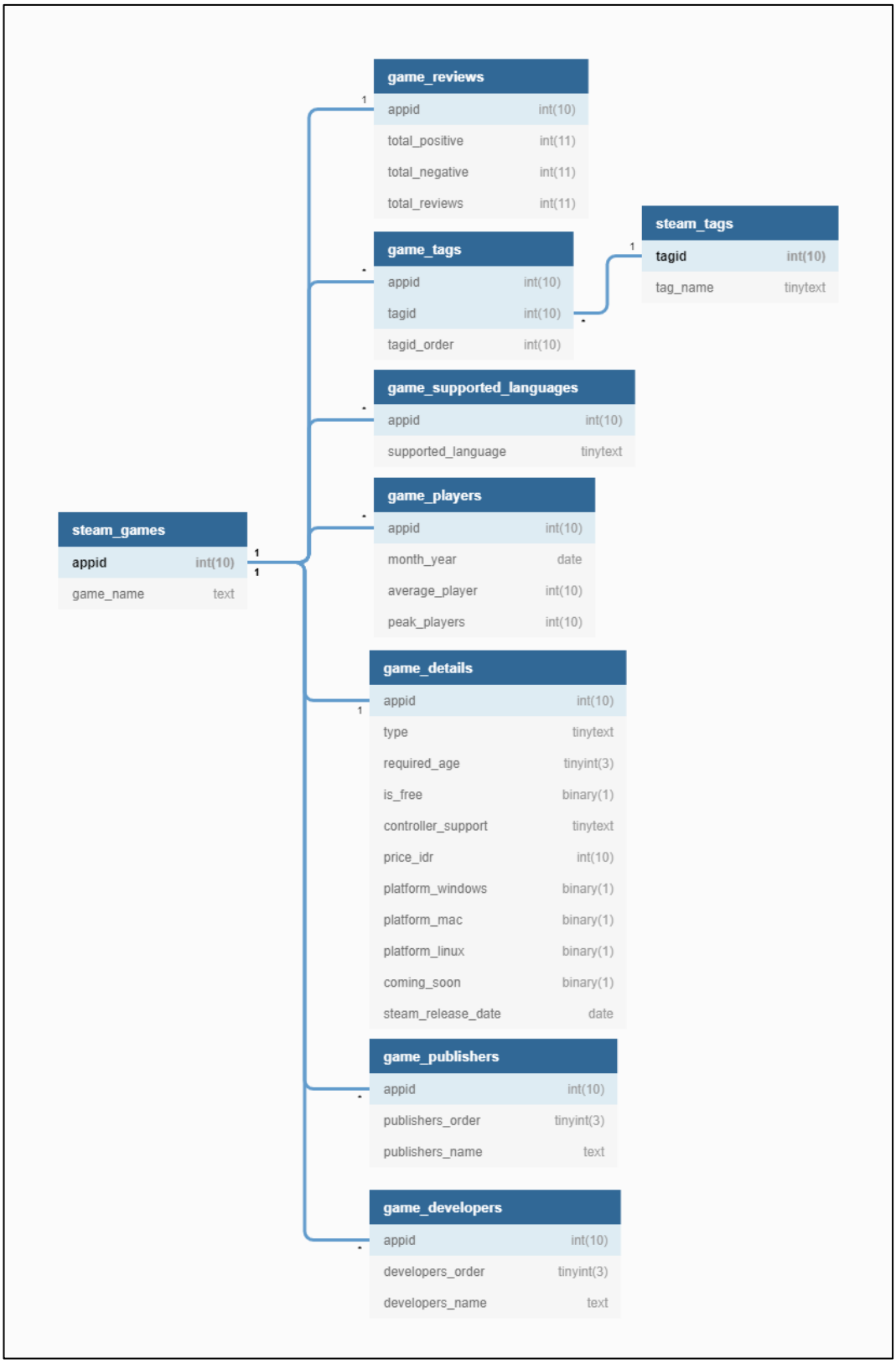
Tabel 3.9 adalah struktur tabel steam_tags yang berfungsi untuk menyimpan data daftar Steam Tags.

Tabel 3.9 Struktur Tabel steam_tags

Nama Kolom	Tipe	Panjang	Keterangan
tagid	int unsigned	10	*Primary Key*
tag_name	tinytext	-	nama Steam Tags

3.4.13 Skema Database Data Game

Gambar 3.5 merupakan skema *database* data *game* secara keseluruhan beserta relasi antar tabel.



Gambar 3.5 Skema *database* data *game*

3.4.14 Desain Antarmuka

Berikut merupakan desain antarmuka aplikasi web untuk pengguna melakukan interaksi dengan aplikasi web.

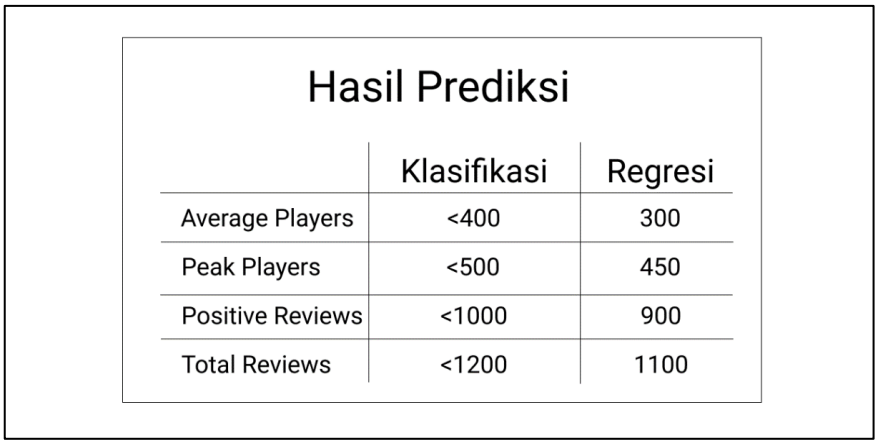
A. Halaman *form* prediksi

Gambar 3.6 merupakan desain antarmuka halaman *form* prediksi yang pengguna isi dengan memilih salah satu opsi dari menu *dropdown* dan mencentang opsi sesuai dengan data *game* yang ingin dibuat.

Gambar 3.6 Desain antarmuka halaman *form* prediksi

B. Halaman hasil prediksi

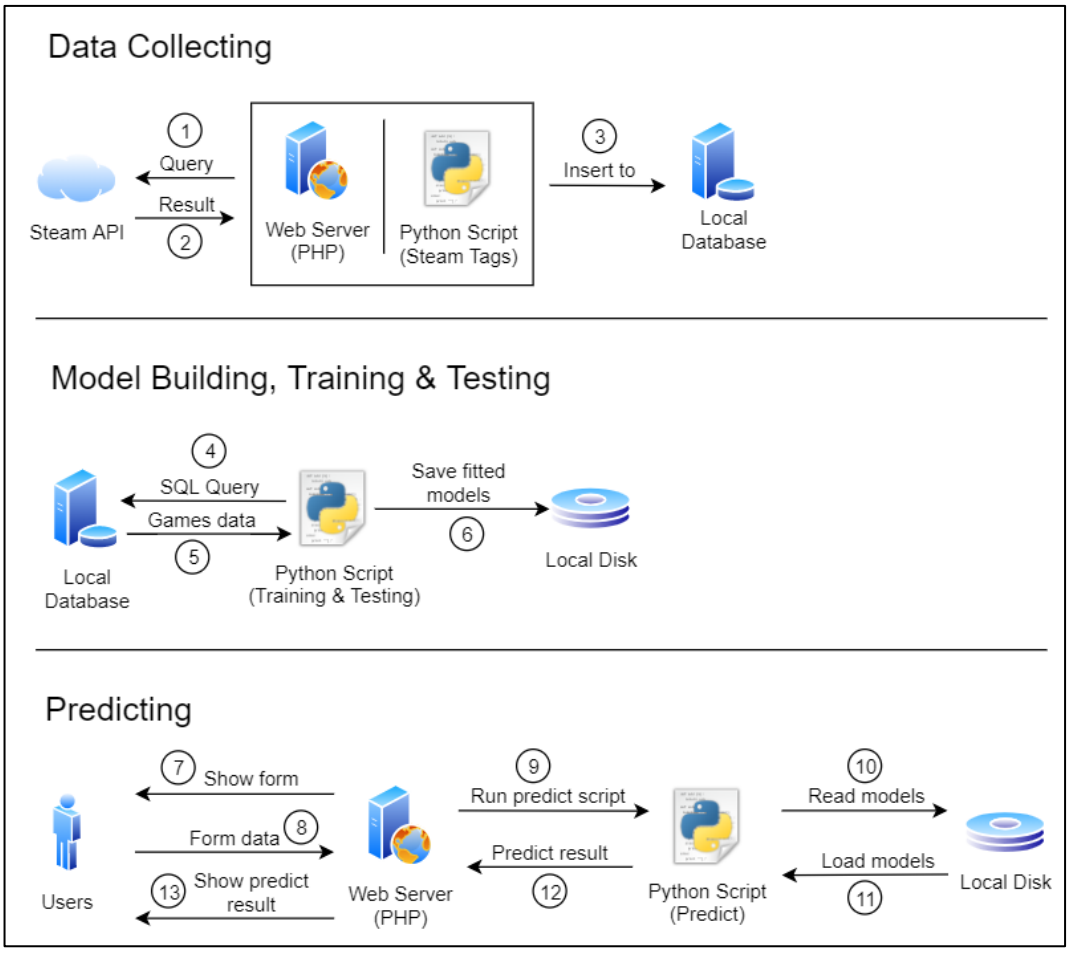
Gambar 3.7 merupakan desain antarmuka halaman hasil prediksi dimana pengguna dapat melihat hasil prediksi untuk *average players*, *peak players*, jumlah ulasan positif, dan jumlah total ulasan.



Gambar 3.7 Desain antarmuka halaman hasil prediksi

3.4.15 Diagram Infrastruktur Aplikasi

Gambar 3.8 merupakan diagram infrastruktur aplikasi ketika melakukan pengumpulan data, pembangunan, pelatihan, dan pengujian model, dan melakukan prediksi.



Gambar 3.8 Diagram infrastruktur aplikasi

3.5 Pengujian Aplikasi

Berikut merupakan penjelasan metrik pengujian aplikasi untuk model klasifikasi dan regresi.

3.5.1 Model Klasifikasi

Berikut merupakan penjelasan metrik pengujian aplikasi untuk model klasifikasi, yaitu: *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Ada beberapa tipe pembeda untuk metrik klasifikasi yaitu:

Tabel 3.10 *Confusion Matrix* untuk klasifikasi biner (Hossin & Sulaiman, 2015)

	Actual Positive	*Actual Negative*
Predicted Positive	*True Positive (TP)*	*False Negative (FN)*
Predicted Negative	*False Positive (FP)*	*True Negative (TN)*

A. Accuracy

Accuracy mengukur rasio prediksi benar dengan total hasil (Hossin & Sulaiman, 2015).

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad \text{Accuracy (3.1)}$$

B. Precision

Precision digunakan untuk mengukur positif yang benar diprediksi dari total prediksi dari kelas positif (Hossin & Sulaiman, 2015).

$$Precision = \frac{TP}{TP+FP} \quad \text{Precision (3.2)}$$

C. Recall

Recall digunakan untuk mengukur positif yang benar diklasifikasi (Hossin & Sulaiman, 2015).

$$Recall = \frac{TP}{TP+TN} \quad \text{Recall (3.3)}$$

D. F1-Score

F1-Score merepresentasikan *harmonic mean* dari nilai *recall* dan *precision* (Hossin & Sulaiman, 2015).

$$F_1 = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad \text{F1-Score (3.4)}$$

3.5.2 Model Regresi

Berikut merupakan penjelasan metrik pengujian aplikasi untuk model regresi, yaitu: *Mean Absolute Error*, dan *Root Mean Square Error*. Ada beberapa notasi untuk metrik regresi, yaitu:

Tabel 3.11 Notasi untuk metrik regresi (Wang & Lu, 2018)

Notasi	Arti Notasi
$\hat{r}_n$	Nilai yang diprediksi
r_n	Nilai yang sebenarnya
N	Jumlah *dataset*

A. Mean Absolute Error

Mean Absolute Error adalah rata-rata perbedaan antara nilai prediksi dengan nilai sebenarnya, dengan kata lain adalah rata-rata *error* prediksi (Tesfamariam & Liu, 2013).

$$MAE = \frac{\sum_{n=1}^{N} |\hat{r}_n - r_n|}{N} \quad \text{Mean Absolute Error (3.5)}$$

B. Root Mean Square Error

Root Mean Square Error merepresentasikan standar deviasi dari perbedaan antara nilai prediksi dengan nilai sebenarnya (Adetiloye & Awasthi, 2017).

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N} (\hat{r}_n - r_n)^2}{N}} \quad \text{Root Mean Square Error (3.6)}$$